L Number	Hits	Search Text	DB	Time stamp
1	144719	(test adj access adj port) or "TAP" or JTAG or (joint adj test adj action adj group)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 06:16
8	5	((test adj access adj port) or "TAP" or JTAG or (joint adj test adj action adj group)) same ((stop or stopp$3 or idle$1 or tristate$1) near3 ((system or bus) adj clock$1))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 06:32
15	0	((test adj access adj port) or "TAP" or JTAG or (joint adj test adj action adj group)) same (ceas$3 near3 bus near2 access)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 06:25
22	3523	((test adj access adj port) or "TAP" or JTAG or (joint adj test adj action adj group)) same (stop$1 or stopp$3 near3 bus near2 access)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 06:25
29	0	((test adj access adj port) or "TAP" or JTAG or (joint adj test adj action adj group)) same ((stop$1 or stopp$3) near3 bus near2 access)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 06:26
36	0	((test adj access adj port) or "TAP" or JTAG or (joint adj test adj action adj group)) and ((stop$1 or stopp$3) near3 bus near2 access)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 06:27
43	6	((test adj access adj port) or "TAP" or JTAG or (joint adj test adj action adj group)) and ((block$3) near3 bus near2 access)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 06:30
50	2	((test adj access adj port) or "TAP" or JTAG or (joint adj test adj action adj group)) and ((ceas$3) near3 bus near2 access)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 06:31
57	28	((test adj access adj port) or "TAP" or JTAG or (joint adj test adj action adj group)) and ((stop or stopp$3 or idle$1 or tristate$1) near3 ((system or bus) adj clock$1)) and (bus near2 access$3)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 06:35
64	0	((test adj access adj port) or "TAP" or JTAG or (joint adj test adj action adj group)) and (((stop or stopp$3 or idle$1 or tristate$1) near3 ((system or bus) adj clock$1)) same (bus near2 access$3))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 06:36
71	7	(debug$4 or (trouble adj shoot$3) or diagnos$4) and (((stop or stopp$3 or idle$1 or tristate$1) near3 ((system or bus) adj clock$1)) same (bus near2 access$3))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 06:42
78	1381	714/$.ccls. and ((test adj access adj port) or "TAP" or JTAG or (joint adj test adj action adj group))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 07:06
85	172	pipeline adj bus	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 08:52
92	14	advantage$3 same (pipeline adj bus)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 07:14

09/705,487

99	0	(vector adj processor$1) same (pipeline adj bus)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 07:15
106	2	(vector adj processor$1) and (pipeline adj bus)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 07:16
113	0	5812562.pn. and (pipeline adj bus)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 07:17
120	0	5812562.pn. and pipeline$1	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 08:34
127	0	(system adj 'on' adj chip) same (pipeline adj bus)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 08:50
134	1	((system adj 'on' adj chip) or system-on-chip) and (pipeline adj bus)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 08:52
141	1099	pipeline$1 near2 bus	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 08:52
148	14	((system adj 'on' adj chip) or system-on-chip) and (pipeline$1 near2 bus)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 08:54
155	844	((system adj 'on' adj chip) or system-on-chip or soc) and pipeline$1	USPAT; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 08:54
161	42	((system adj 'on' adj chip) or system-on-chip or soc) same pipeline$1	USPAT; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 08:55
167	1	((system adj 'on' adj chip) or system-on-chip) same pipeline$1	USPAT; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 08:57
173	0	((system adj 'on' adj chip) or system-on-chip) and (pipeline$1 near2 bus)	USPAT; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 08:58
179	4	(DSP or MSP) same (pipeline$1 near2 bus)	USPAT; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 08:58
-	112	debug$4 same (trouble adj shoot$3)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:24
-	22	(port$1 or interface$1) same (debug$4 same (trouble adj shoot$3))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:24

-	2	JTAG same ((port$1 or interface$1) same (debug$4 same (trouble adj shoot$3)))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:24
-	0	(debug$4 same (trouble adj shoot$3)) and (hardware near2 state$1 near3 static)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:15
-	4	hardware near2 state$1 near3 static	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:19
-	587	(stop or stopp$3 or idle$1) near3 ((system or bus) adj clock$1)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:22
-	0	(debug$4 same (trouble adj shoot$3)) same ((stop or stopp$3 or idle$1) near3 ((system or bus) adj clock$1))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:22
-	1	(debug$4 same (trouble adj shoot$3)) and ((stop or stopp$3 or idle$1) near3 ((system or bus) adj clock$1))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:22
-	25967	debug$4 or (trouble adj shoot$3)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:24
-	5142	(port$1 or interface$1) same (debug$4 or (trouble adj shoot$3))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:24
-	336	JTAG same ((port$1 or interface$1) same (debug$4 or (trouble adj shoot$3)))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:24
-	3	JTAG same ((stop or stopp$3 or idle$1) near3 ((system or bus) adj clock$1))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/30 06:17

US-PAT-NO: 5479635

DOCUMENT-IDENTIFIER: US 5479635 A

TITLE: Memory device including DRAMs for high-speed accessing

DATE-ISSUED: December 26, 1995

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Kametani; Masatsugu	Ibaraki	N/A	N/A	JP

US-CL-CURRENT: 711/5, 365/222 , 365/238.5 , 711/105 , 711/106 , 711/209

ABSTRACT:

A memory device comprises a dynamic random access memory (DRAM) organized by page and a memory access devices. The DRAM corresponding to the pages is divided into a plurality of groups each constituted of pages for storing data which are unlikely to give rise to interference between pages. The DRAM of each group is constituted as a memory system which responds to page access. The memory access devices are provided separately for the memory system of each group. Each memory access device has a memory means which, in response to an access designating a page address of the memory system associated therewith, stores an old page address designated at least one access earlier, and judging means which, in response to said page address access, judges whether or not the new page address designated by said access coincides with said old page address stored in said storage means. Page access is conducted in accordance with the old address if the judging means judges that the old and new page addresses coincide and is conducted in accordance with the new address after changing the page to be accessed to said new page if the old and new page addresses do not coincide.

15 Claims, 14 Drawing figures

Exemplary Claim Number: 1

Number of Drawing Sheets: 9

---------- KWIC ---------

Detailed Description Text - DETX (51):

The operation and effect of this embodiment will be better understood from FIG. 13(A) and 13(B). FIG. 13(B) shows the situation in the case of ordinary page access (in high-speed page mode) according to this invention. BS denotes the processor bus state and in this embodiment one bus cycle is equal to one processor cycle. Further, this embodiment relates to an application using a processor which conducts bus access using **pipeline bus** access. **Pipeline bus** access is an access system in which the address to be used in a given bus cycle is output in the preceding bus cycle (or processor cycle). The output address is latched and then used in the actual bus cycle. Use of this method makes it possible to take full **advantage** of the bus cycle time in carrying out access. With progressive reduction of processor machine cycle time, it will become increasingly difficult to secure adequate address access time when the address is output in the same bus cycle and because of this, it is expected that more and more processors will be employing **pipeline bus** cycle accessing in which the addresses are output within the preceding cycle. In the present invention,

instructions and data which are unlikely to give rise to inter-page interference are clumped together in separate groups. For example, instruction data is put in one group and array variable data in another group, so that in each group there is a high probability of consecutive data being located in consecutive address space. More specifically, when the processor accesses the dynamic memory system to which a given group has been assigned, the probability is high that the even and odd numbered addresses will appear alternately. In the ordinary page accessing illustrated in FIG. 13(B), it is necessary within each bus cycle BS to secure a CAS signal precharge time (CAS precharge time) pt2 and an access time at2 during which CAS is kept active. The problem is that after the time pt2 has been secured, the time at2 cannot be made as long as required. This problem can be overcome by using the 2-bank system according to the present invention illustrated in FIG. 12. The time chart shown in FIG. 13(A) relates to page access according to the present embodiment. In a bus state having an even data address (m and n in the figure being integers), CASE 160 is made active whereas in a bus state having an odd data address, CASo 161 is made active. CASE 160 makes bank E300 of the dynamic memory system active whereas CASo makes bank o301 active. Since even- and odd-numbered addresses appear alternately from BS.sub.2n-1 to BS.sub.2n+2, CASE 160 and CASo 161 are alternately active. Thus the periods when the respective banks are not being accessed can be used for securing the precharge time pt1 for CASE and CASo. However, after the switchover of the data addresses to BS.sub.2n+2 and BS.sub.2m, even addresses (2n+2 and 2m) occur in succession. Thus since the same bank (bank E300) is accessed consecutively, it is liable to be impossible to secure the required precharge time. This problem is coped with by the bank selection means 13b, which discriminates when consecutive accessing of the same bank occurs and then generates and sends to the RAS/CAS generator 13 a WAIT signal 7C. Based on the WAIT signal 7C and the CAS generation signal 16, the RAS/CAS generator 13 informs the processor (the READY signal generator 22 in the case of the example of FIG. 1) over a signal line 24a so as to have a WAIT state inserted in the bus cycle concerned. As a result, a WAIT state BSW.sub.2m is introduced. Then after the CASE 160 precharge time has been secured by BSW.sub.2m, CASE 160 is made active by BSW.sub.2m, with the result that successive accessing of bank E300 can be carried out with no inconsistency. In this connection, even where consecutive access of the same bank occurs, it is more effective to arrange for the WAIT signal 7C not to be generated in a case where one or more idle states (meaning a cycle in which the processor does not request access to the dynamic memory system 3 concerned) have been introduced between the two bus cycles. In the case where the bank CAS generating means 13a receives CAS switch information 7b and the CAS generation signal from the RAS/CAS generator 13 is active, if signal 7b instructs access of the bank E300, CASE 160 is made active LO and applied to the bank E300, and if the signal 7b instructs access of the bank o301, CASo 160 is made active LO and applied to the bank o301.

US-PAT-NO: 5787025

DOCUMENT-IDENTIFIER: US 5787025 A

TITLE: Method and system for performing arithmetic operations with single or double precision

DATE-ISSUED: July 28, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Muwafi; Jumana A.	San Francisco	CA	N/A	N/A
Touriguian; Mihran	Hercules	CA	N/A	N/A

US-CL-CURRENT: 708/490, 708/231 , 708/233

ABSTRACT:

A circuit for performing either single precision or double precision arithmetic operations on data, a system including such a circuit, and a method implemented by the system. Preferably, the circuit is an arithmetic manipulation unit (AMU) which performs arithmetic operations on N-bit words in a single precision mode and on 2N-bit words in a double precision mode. The AMU concatenates two N-bit words in the double precision mode thus producing a 2N-bit operand, and performs a selected one of several arithmetic operations on the operand and a second 2N-bit operand. Preferably, the AMU performs a double precision operation in two cycles: a first cycle generating a first operand and loading the operand to an output register; and a second cycle in which a second operand is generated from a second pair of N-bit parts from the memory, the first operand is fed back from the output register, and an arithmetic operation is performed on the two operands. The system preferably includes a multi-port memory, executes instructions in pipelined fashion, and operates in a single precision mode to fetch two N-bit operands from the memory in a single pipeline cycle using two address pointers and in a double precision mode to fetch two N-bit words from the memory in a single cycle. In a double precision mode, the system can fetch two N-bit parts stored in consecutive memory locations in a single cycle, by generating and asserting to the memory two addresses in response to one address pointer.

23 Claims, 9 Drawing figures

Exemplary Claim Number: 1

Number of Drawing Sheets: 8

---------- KWIC ---------

Detailed Description Text - DETX (30):

Preferably, the **DSP** is programmed to execute instructions in a pipelined manner (in which addresses for reads from a memory are asserted on an address **bus in one pipeline** cycle and addresses for writes to the memory are asserted on the same address **bus in a different pipeline** cycle), and in a manner avoiding expected address bus and data bus conflicts or collisions.

without exposing the code stream to possible faults in the computer system's main memory.

Detailed Description Text - DETX (305):
The I/O device 26 then writes a block of **diagnostic** code to the same address space to which the backup cache was initialized. The addresses and corresponding data are accepted by the backup cache 226 in accordance with the implemented update v. invalidate policy. The backup cache 226 updates the contents of the cache entries with the **diagnostic** code being supplied via the system bus writes initiated by the I/O device 26. As the backup cache entries are updated with the **diagnostic** code, the corresponding DIRTY and VALID status bits for each entry will remain asserted since the force TAG status control bit remains asserted.

Detailed Description Text - DETX (306):
Once the **diagnostic** code is written to the backup cache 226 in the above manner, future system bus reads to the address locations will result in the cache contents being supplied in response to any read operation, in accordance with the system bus protocol, since the cache entries are marked DIRTY. Furthermore, in accordance with the implemented write through cache policy, of the primary caches, future write operations by the processor 202 to an address located in the backup cache 226 will result in the backup cache entry being updated.

Detailed Description Text - DETX (308):
The above mechanism, for permitting the backup caches to mimic main memory, permits code to be executed without being exposed to main memory hardware faults. Such a mechanism, which permits **diagnostic** code to be stored in non-volatile RAM on a slow computer subsystem, e.g. the I/O device 26, and yet permits the **diagnostic** code to be executed without exposure to main memory system hardware faults, provides significant advantages over the known systems which store the **diagnostic** code on the CPU module or which require operation of the computer systems main memory to permit execution of **diagnostic** code.

Detailed Description Text - DETX (402):
The bus arbitration logic block 256 of the odd slice 236 of the primary CPU module's bus interface unit 232 receives requests to **access the system bus** 28, from the various modules coupled to the system bus 28. **Access to the system bus** 28 is granted by the bus arbitration logic block 256, to a requesting module, in accordance with a bus arbitration scheme that is intended to maximize overall system performance, as will be described below. In addition to having the ability to grant **access to the system bus** 28, the bus arbitration logic block 256 of the odd slice 236 of the primary CPU module, has the capability of inserting one or more idle bus cycles, where each **idle bus cycle comprises a bus clock** cycle during which no module on the system bus 28 is granted **access to the system bus** 28.

L Number	Hits	Search Text	DB	Time stamp
-	112	debug$4 same (trouble adj shoot$3)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:24
-	22	(port$1 or interface$1) same (debug$4 same (trouble adj shoot$3))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:24
-	2	JTAG same ((port$1 or interface$1) same (debug$4 same (trouble adj shoot$3)))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:24
-	0	(debug$4 same (trouble adj shoot$3)) and (hardware near2 state$1 near3 static)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:15
-	4	hardware near2 state$1 near3 static	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:19
-	587	(stop or stopp$3 or idle$1) near3 ((system or bus) adj clock$1)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:22
-	0	(debug$4 same (trouble adj shoot$3)) same ((stop or stopp$3 or idle$1) near3 ((system or bus) adj clock$1))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:22
-	1	(debug$4 same (trouble adj shoot$3)) and ((stop or stopp$3 or idle$1) near3 ((system or bus) adj clock$1))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:22
-	25967	debug$4 or (trouble adj shoot$3)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:24
-	5142	(port$1 or interface$1) same (debug$4 or (trouble adj shoot$3))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:24
-	336	JTAG same ((port$1 or interface$1) same (debug$4 or (trouble adj shoot$3)))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:24
-	3	JTAG same ((stop or stopp$3 or idle$1) near3 ((system or bus) adj clock$1))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/29 20:24

US-PAT-NO: 5793776

DOCUMENT-IDENTIFIER: US 5793776 A

TITLE: Structure and method for SDRAM dynamic self refresh entry and exit using JTAG

DATE-ISSUED: August 11, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Qureshi; Amjad	San Jose	CA	N/A	N/A
Baeg; Sanghyeon	Cupertino	CA	N/A	N/A

US-CL-CURRENT: 714/724, 714/42

ABSTRACT:

JTAG test logic and a memory controller place an SDRAM in a self refresh mode prior to beginning JTAG testing. The memory controller can complete a current memory access and otherwise prepare for the JTAG test. During the JTAG test, self refresh mode operation of the SDRAM retains data without the need for a clock signal or refresh signals which are suspended for the JTAG test. Accordingly, after the JTAG test, circuit operation can continue without reinitializing data in the SDRAM.

16 Claims, 7 Drawing figures

Exemplary Claim Number: 1

Number of Drawing Sheets: 6

---------- KWIC ---------

Brief Summary Text - BSTX (11):

A **JTAG** Logic is used in accordance with this invention to asynchronously communicate with a Memory Controller Unit to allow the **stopping of system clocks** while preserving the contents of SDRAM using the self refresh mode. The Memory Controller Unit does not permit the **system clocks to be stopped by the JTAG** Logic for testing until the Memory Controller Unit has finished the current memory access operation. Prior to the **stopping of the system clocks,** the Memory Controller Unit places the SDRAM into self refresh mode to preserve the memory contents.

Brief Summary Text - BSTX (13):

In one embodiment in accordance with this invention, when the **JTAG** Controller wants to **stop the system clock** to allow testing to commence, a signal jtag.sub.-- clk.sub.-- stop.sub.-- req high is asserted and communicated via a Memory Control Register to a Self-Refresh State Machine which is part of the Memory Controller Unit. The Self-Refresh State Machine asserts the signal jtag.sub.-- clk.sub.-- stop.sub.-- request high to a Memory Controller State Machine which finishes the current memory access operation before asserting a signal mcu.sub.-- idle high back to the Self-Refresh State Machine. On assertion of the signal mcu.sub.-- idle high by the Memory Controller State Machine and the presence of signal jtag.sub.-- clk.sub.-- stop.sub.-- req high, the Self-Refresh State Machine places the SDRAM into self refresh mode. The

Self-Refresh State Machine also asserts the signal mcu.sub.-- idle high to an Observation Control Register which is continually scanned by the **JTAG** Controller. If the **JTAG** Controller detects the signal mcu.sub.-- idle high and the signal jtag.sub.-- clk.sub.-- stop.sub.13 req high, a signal sys.sub.13 clk.sub.-- bypass high is asserted by the **JTAG** Controller. The signal sys.sub.-- clk.sub.-- bypass high is asserted to a System Clock Generator Block via the Memory Control Register and causes the system clock to be bypassed.

US-PAT-NO: 5629950

DOCUMENT-IDENTIFIER: US 5629950 A

TITLE: Fault management scheme for a cache memory

DATE-ISSUED: May 13, 1997

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Godiwala; Nitin D.	Boylston	MA	N/A	N/A
Thaller; Kurt M.	Acton	MA	N/A	N/A
Metzger; Jeffrey A.	Leominster	MA	N/A	N/A
Maskas; Barry A.	Sterling	MA	N/A	N/A

US-CL-CURRENT: 714/805, 711/131 , 711/144 , 714/764

ABSTRACT:

The present invention is directed to a method of managing a cache upon detection of an address TAG parity error, The cache includes a plurality of entries for storage of data, with each entry having a corresponding address TAG entry. The method includes the steps of performing a TAG parity check for each access to the cache, and upon detection of a parity error in an address TAG, disabling allocation of TAG entries for storage of new address TAGs. A signal indicating the TAG parity error is transmitted to an error correction mechanism.

10 Claims, 9 Drawing figures

Exemplary Claim Number: 1

Number of Drawing Sheets: 9

---------- KWIC ---------

Detailed Description Text - DETX (11):

The I/O module 26 of the computer system 10 comprises a system bus interface unit 134, which is coupled to a series of other devices contained within the I/O module 26, including a disk drive interface 118, a CONSOLE serial line interface TOY (time of year) clock 130, a CONSOLE FLASH EPROM ("FEPROM") 124 and a FUTUREBUS plus interface block 132. The console FEPROM 124 may be used to store **diagnostic** and initialization data for use by the systems CPU modules 14, 16, as will be described below. In order to facilitate system performance, the I/O device 26 may further comprise a processor 156 and a cache 150 coupled to the I/O module's system bus interface unit 134.

Detailed Description Text - DETX (289):

This ability, to initialize the backup cache 226 and its contents to any desired state, may be used for initialization purposes or by **diagnostic** programs as will appear.

Detailed Description Text - DETX (299):

The mimic main memory ability of the computer system 10 is used by the computer system 10 in the following manner, to load and execute **diagnostic** code

without exposing the code stream to possible faults in the computer system's main memory.

Detailed Description Text - DETX (306):
The I/O device 26 then writes a block of **diagnostic** code to the same address space to which the backup cache was initialized. The addresses and corresponding data are accepted by the backup cache 226 in accordance with the implemented update v. invalidate policy. The backup cache 226 updates the contents of the cache entries with the **diagnostic** code being supplied via the system bus writes initiated by the I/O device 26. As the backup cache entries are updated with the **diagnostic** code, the corresponding DIRTY and VALID status bits for each entry will remain asserted since the force TAG status control bit remains asserted.

Detailed Description Text - DETX (307):
Once the **diagnostic** code is written to the backup cache 226 in the above manner, future system bus reads to the address locations will result in the cache contents being supplied in response to any read operation, in accordance with the system bus protocol, since the cache entries are marked DIRTY. Furthermore, in accordance with the implemented write through cache policy, of the primary caches, future write operations by the processor 202 to an address located in the backup cache 226 will result in the backup cache entry being updated.

Detailed Description Text - DETX (309):
The above mechanism, for permitting the backup caches to mimic main memory, permits code to be executed without being exposed to main memory hardware faults. Such a mechanism, which permits **diagnostic** code to be stored in non-volatile RAM on a slow computer subsystem, e.g. the I/O device 26, and yet permits the **diagnostic** code to be executed without exposure to main memory system hardware faults, provides significant advantages over the known systems which store the **diagnostic** code on the CPU module or which require operation of the computer systems main memory to permit execution of **diagnostic** code.

Detailed Description Text - DETX (397):
The bus arbitration logic block 256 of the odd slice 236 of the primary CPU module's bus interface unit 232 receives requests to **access the system bus** 28, from the various modules coupled to the system bus 28. **Access to the system bus** 28 is granted by the bus arbitration logic block 256, to a requesting module, in accordance with a bus arbitration scheme that is intended to maximize overall system performance, as will be described below. In addition to having the ability to grant **access to the system bus** 28, the bus arbitration logic block 256 of the odd slice 236 of the primary CPU module, has the capability of inserting one or more idle bus cycles, where each **idle bus cycle comprises a bus clock** cycle during which no module on the system bus 28 is granted **access to the system bus** 28.

US-PAT-NO: 5555382

DOCUMENT-IDENTIFIER: US 5555382 A

TITLE: Intelligent snoopy bus arbiter

DATE-ISSUED: September 10, 1996

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Thaller; Kurt M.	Acton	MA	N/A	N/A
Godiwala; Nitin D.	Boylston	MA	N/A	N/A
Maskas; Barry A.	Sterling	MA	N/A	N/A

US-CL-CURRENT: 710/113, 710/100 , 710/119 , 711/131 , 711/146

ABSTRACT:

The present invention is directed to a method for arbitrating for control of a bus in a multiprocessor system. The multiprocessor system comprises a plurality of processors and a main memory coupled to one another by the bus, each processor including a cache memory accessible by the corresponding processor and in connection with transactions on the bus. The method includes the steps of generating requests for control of the bus and granting control of the bus in respect of one of the requests. The bus is monitored for preselected transaction activity on the bus; and an idle cycle is inserted on the bus upon monitoring the preselected transaction activity.

5 Claims, 11 Drawing figures

Exemplary Claim Number: 1

Number of Drawing Sheets: 9

---------- KWIC ---------

Detailed Description Text - DETX (11):
The I/O module 26 of the computer system 10 comprises a system bus interface unit 134, which is coupled to a series of other devices contained within the I/O module 26, including a disk drive interface 118, a CONSOLE serial line interface TOY (time of year) clock 130, a CONSOLE FLASH EPROM ("FEPROM") 124 and a FUTUREBUS plus interface block 132. The console FEPROM 124 may be used to store **diagnostic** and initialization data for use by the systems CPU modules 14, 16, as will be described below. In order to facilitate system performance, the I/O device 26 may further comprise a processor 156 and a cache 150 coupled to the I/O module's system bus interface unit 134.

Detailed Description Text - DETX (288):
This ability, to initialize the backup cache 226 and its contents to any desired state, may be used for initialization purposes or by **diagnostic** programs as will appear.

Detailed Description Text - DETX (298):
The mimic main memory ability of the computer system 10 is used by the computer system 10 in the following manner, to load and execute **diagnostic** code